Exploring Recommendation Systems Algorithms and its metrics

1. Abstract

In this project, I have explored the algorithms and evaluation parameters utilized by popular recommendation systems, understand the purpose of using the specific algorithms for different type of dataset such as text reviews, rating-based recommendation system as well as ML agents and evaluate these parameters on different metrics. For this purpose, we can build recommendation system based on Machine learning, Deep Learning and Reinforcement Learning algorithms and evaluate them using metrics such as Predictive accuracy metrics, classification accuracy metrics, rank accuracy metrics, Accumulative reward, and non-accuracy measurements.

I would like this project to serve as a tutorial as well as a comparison survey to analyze how a particular recommendation system technique can be simulate based on the dataset and how to choose a metrics which would be appropriate to evaluate such models. Can we generalize any of these approaches?

2. Introduction

2.1. Overview of Recommendation Systems

Recommender systems are used in a variety of areas, with commonly recognized examples taking the form of playlist generators for video and music services, product recommenders for online stores, or content recommenders for social media platforms and open web content recommenders. These systems can operate using a single input, like music, or multiple inputs within and across platforms like news, books, and search queries. There are also popular recommender systems for specific topics like restaurants and online dating. Recommender systems have also been developed to explore research articles and experts, collaborators, and financial services.

A recommendation system seeks to understand the user preferences with the objective of recommending him or her items. These systems have become increasingly popular in recent years, in parallel with the growth of internet retailers like Amazon, Netflix or Spotify. Recommender systems are used in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. In terms of business impact, according to a recent study from Wharton School, recommendation engines can cause a 25% lift in number of views and 35% lift in number of items purchased. So, it is worth to understand these systems.

Recommendation systems (RS) are omnipresent phenomenon which we encounter in our day-to-day life. Powering the modern-day e-commerce systems are gigantic ‘recommendation engines’, looking at each individual transactions, mapping customer profiles and using them to recommend personalized products and services.

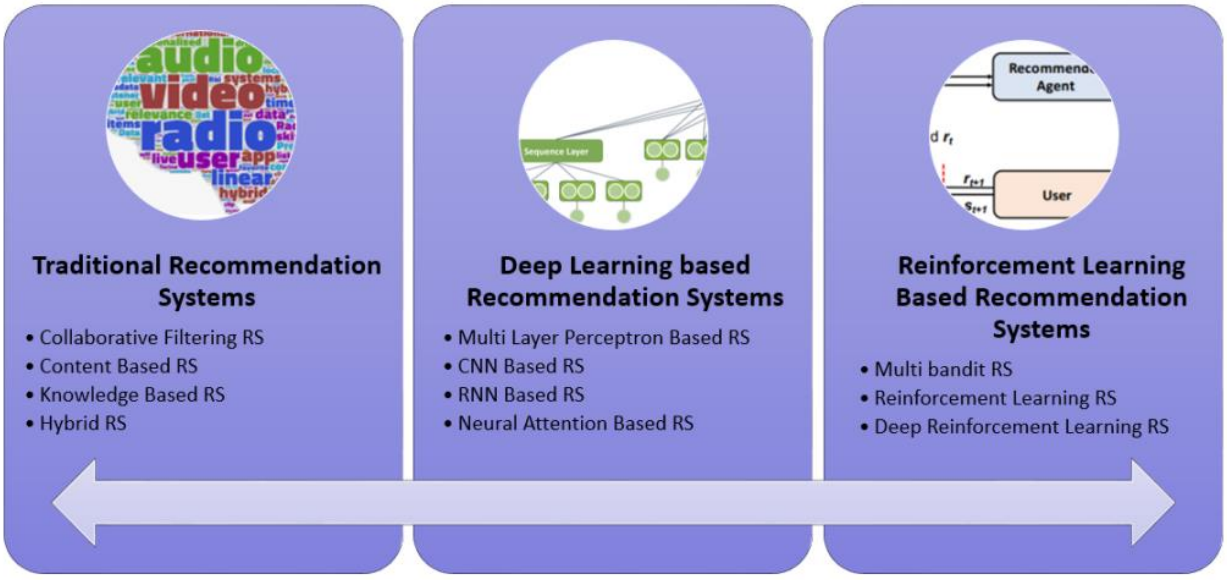
2.2. Evolution of Recommendation Systems

Recommendation Systems	Products
Amazon	Books, electronics, consumer goods and other products
Facebook	Friends, advertisements and other services
Youtube	Online videos
Netflix	Streaming video
MovieLens	Movies
Google Search	Online advertisements
Google news	News articles
Tripadvisor	Travel services / products

3. Datasets

- Books - http://www2.informatik.uni-freiburg.de/~cziegler/BX/
- Movie lens - http://labrosa.ee.columbia.edu/millionsong/
- Amazon electronics - https://jmcauley.ucsd.edu/data/amazon/
- Netflix prize - https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data

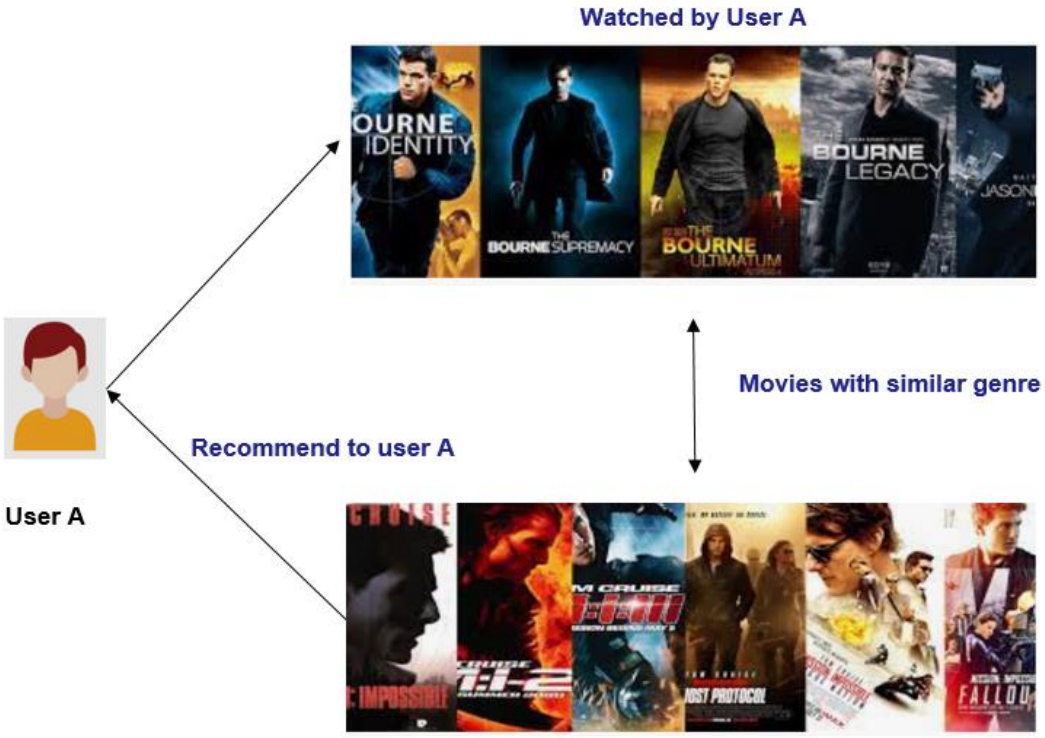
4. Types of Recommendation Systems



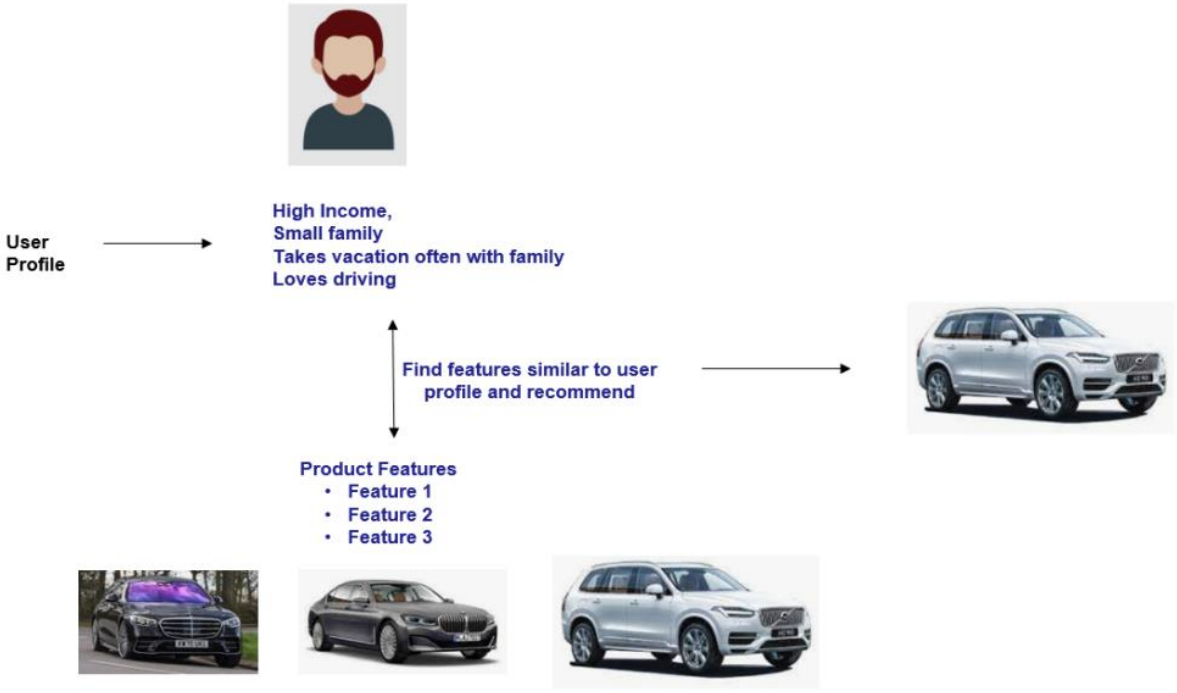
1. **Collaborative filtering** collects large amounts of information on users' behaviors, activities, or preferences to predict what users will like based on their similarity to other users. This information can be explicit, where the user directly provides the ratings of the items, or implicit, where the ratings must be extracted for the implicit user behavior, like number of views, likes, purchases, etc.




2. **Content-based filtering** consider contextual user factors such as location, date of purchase, user demographics and item factors like price, brand, type of item, etc., to recommend items that are like those that a user liked in the past. The system creates a content-based profile of users based on a weighted vector of item features. The weights denote the importance of each feature to the user and can be computed from individually rated content vectors using a variety of techniques. Simple approaches use the average values of the rated item vector while other sophisticated methods use machine learning techniques such as Bayesian Classifiers, cluster analysis, decision trees, and neural networks to estimate the probability that the user is going to like the item.



3. **Knowledge Based Recommendation Systems**: These types of systems make recommendations based on similarity between a user's requirements and an item description. Knowledge based recommendation systems are usually useful in context where the purchases infrequent like buying an automobile, real estate, luxury goods etc.



4. **Hybrid recommender systems** combine multiple techniques together to achieve some synergy between them. They can use aspects from collaborative filtering, content-based filtering, knowledge based and demographics. They have proved to be very effective in some cases, like the Bellkor solution, winner of the Netflix prize, the Netflix Recommender System or Light FM.

5. Methodology of Recommendation Systems

In this project I have implemented below recommendation systems on various text datasets. Each of this notebook would also serve as a tutorial in implementing this type of system.

5.1. Traditional Collaborative filtering for Books, movie lens and amazon electronics dataset

5.1.1. Singular Value Decomposition (SVD)

The neighborhood model is an item-oriented approach to discover the user preference based on the ratings given by the user for similar items. On the other hand, latent factor models such as Singular Value Decomposition (SVD) extract features and correlation from the user-item matrix. For example, when items are movies in different categories. SVD would generate factors when looking into the dimension space like action vs comedy, Hollywood vs Bollywood, or Marvel vs Disney.

When it comes to dimensionality reduction, the Singular Value Decomposition (SVD) is a popular method in linear algebra for matrix factorization in machine learning. Such a method shrinks the space dimension from N-dimension to K-dimension (where K<N) and reduces the number of features. SVD constructs a matrix with the row of users and columns of items and the elements are given by the users' ratings. Singular value decomposition decomposes a matrix into three other matrices and extracts the factors from the factorization of a high-level (user-item-rating) matrix.

$$A = USV^T$$

Matrix U: singular matrix of (user*latent factors)
Matrix S: diagonal matrix (shows the strength of each latent factor)
Matrix U: singular matrix of (item*latent factors)

From matrix factorization, the latent factors show the characteristics of the items. Finally, the utility matrix A is produced with shape m*n. The final output of the matrix A reduces the dimension through latent factors' extraction. From the matrix A, it shows the relationships between users and items by mapping the user and item into r-dimensional latent space. Vector X_i is considered each item and vector Y_u is regarded as each user. The rating is given by a user on an item as R_ui = X^T_i * Y_u. The loss can be minimized by the square error difference between the product of R_ui and the expected rating.

$$Min(x,\ y) \sum_{(u,\ i) \in K} (r_{ui} - x_i^T \cdot y_u)^2$$

Regularization is used to avoid overfitting and generalize the dataset by adding the penalty.

$$Min(x,\ y) \sum_{(u,\ i) \in K} (r_{ui} - x_i^T \cdot y_u)^2 + \lambda(||x_i||^2 + ||y_u||^2)$$

Here, we add a bias term to reduce the error of actual versus predicted value by the model. The equation below adds the bias term and the regularization term:

$$Min(x,\ y,\ b_i,\ b_u) \sum_{(u,\ i) \in K} (r_{ui} - x_i^T \cdot y_u - \mu - b_i - b_u)^2 + \lambda(||x_i||^2 + ||y_u||^2 + b_i^2 + b_u^2)$$

5.2. Nearest Neighbor item based-collaborative filtering

kNN is a machine learning algorithm to find clusters of similar users based on common book ratings and make predictions using the average rating of top-k nearest neighbors. For

example, we first present ratings in a matrix, with the matrix having one row for each item (book) and one column for each user. We then find the k item that have the most similar user engagement vectors.

After fitting that matrix, k-neighbors can be used for to find out which books/products are Similar. In this By Applying Collaborative filtering using K-Nearest Neighbors, we will convert that 2D Matrix into and fill missing values with zeroes others with given ratings, now we will calculate distances between rating vectors, then we will transform the values(ratings) of matrix dataframe into a scipy sparse for more efficient calculations.

5.3. Popularity based recommendation system for Songs dataset

It is a type of recommendation system which works on the principle of popularity and or anything which is in trend. These systems check about the product or movie which are in trend or are most popular among the users and directly recommend those.

For example, if a product is often purchased by most people, then the system will get to know that that product is most popular so for every new user who just signed it, the system will recommend that product to that user also and chances becomes high that the new user will also purchase that.

Merits of popularity-based recommendation system

- It does not suffer from cold start problems which means on day 1 of the business also it can recommend products on various different filters.
- There is no need for the user's historical data.

Demerits of popularity-based recommendation system

- Not personalized
- The system would recommend the same sort of products/movies which are solely based upon popularity to every other user.

Example

- Google News: News filtered by trending and most popular news.
- YouTube: Trending videos.

5.4. Collaborative filtering with Autoencoders

Once we have the data, let's explain in some detail the model that we are going to use. The model, developed by NVIDIA folks, is a Deep autoencoder with 6 layers with non-linear activation function SELU (scaled exponential linear units), dropout and iterative dense refeeding.

An autoencoder is a network which implements two transformations: $encode(x)$: $Rn \Rightarrow Rd$ and $decoder(z)$:$Rd \Rightarrow Rn$. The "goal" of autoencoder is to obtain a d dimensional representation of data such that an error measure between x and $f(x)$=$decode(encode(x))$ is minimized. In the next figure, the auto coder architecture proposed in the paper is showed. Encoder has 2 layers $e1$ and $e2$ and decoder has 2 layers $d1$ and $d2$. Dropout may be applied to coding layer z. In the paper, the authors show experiments with different number of layers, from 2 to 12 (see Table 2 in the original paper).

During the forward pass the model takes a user representation by his vector of ratings from the training set $x \in Rn$, where n is number of items. Note that x is very sparse, while the output of the decoder, $y=f(x) \in Rn$ is dense and contains the rating predictions for all items in the corpus. The loss is the root mean squared error (RMSE).

One of the key ideas of the paper is dense re-feeding. Let's consider an idealized scenario with a perfect f. Then $f(x)i=xi$, $\forall i: xi \neq 0$ and $f(x)i$ accurately predicts all user's future ratings. This means that if a user rates a new item k (thereby creating a new vector x') then $f(x)k=x'k$ and $f(x)=f(x')$. Therefore, the authors refeed the input in the autoencoder to augment the dataset. The method consists of the following steps:

1. Given a sparse x, compute the forward pass to get $f(x)$ and the loss.
2. Backpropagate the loss and update the weights.
3. Treat $f(x)$ as a new example and compute $f(f(x))$
4. Compute a second backward pass.

Steps 3 and 4 can be repeated several times.

6. Evaluation parameters and Results

Recommender System accuracy is popularly evaluated through two main measures: Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). Both are nice as they allow for easy interpretation: they're both on the same scale as the original ratings. However, one may be better to use, depending on the context of the data set.

6.1. SVD collaborative filtering results

10. Get top 10 recommendation for above given userID

```
In [53]: recommendations.sort_values('Predictions', ascending = False).iloc[:10, :]
```

Out[53]:

	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher	Predictions
407	0316666343	The Lovely Bones: A Novel	Alice Sebold	2002	Little, Brown	1.015397
2116	0345350499	The Mists of Avalon	MARION ZIMMER BRADLEY	1987	Del Rey	0.697309
2438	0440214041	The Pelican Brief	John Grisham	1993	Dell	0.665439
455	044021145X	The Firm	John Grisham	1992	Bantam Dell Publishing Group	0.663549
521	0312195516	The Red Tent (Bestselling Backlist)	Anita Diamant	1998	Picador USA	0.642840
20670	0345318862	Golem in the Gears (Xanth Novels (Paperback))	PIERS ANTHONY	1986	Del Rey	0.639465
4810	0345313151	Bearing an Hourglass (Incarnations of Immortal...	Piers Anthony	1991	Del Rey Books	0.631446
6320	0380752891	Man from Mundania (Xanth Trilogy, No 12)	Piers Anthony	1990	Harper Mass Market Paperbacks	0.629143
44448	051511605X	Undue Influence	Steven Paul Martini	1995	Jove Books	0.617955
8977	043936213X	Harry Potter and the Sorcerer's Stone (Book 1)	J. K. Rowling	2001	Scholastic	0.614288

6.2. Popularity based recommendation system

```
Here is the recommendation for the userId: 15

      userID             song_id  score  Rank
378       15  SOBONKR12A58A7A7E0     34   1.0
209       15  SOAUWYT12A81C206F1     31   2.0
1399      15  SOFRQTD12A81C233C0     30   3.0
223       15  SOAXGDH12A8C13F8A1     24   4.0
4526      15  SOSXLTC12AF72A7F54     23   5.0


Here is the recommendation for the userId: 121

      userID             song_id  score  Rank
378      121  SOBONKR12A58A7A7E0     34   1.0
209      121  SOAUWYT12A81C206F1     31   2.0
1399     121  SOFRQTD12A81C233C0     30   3.0
223      121  SOAXGDH12A8C13F8A1     24   4.0
4526     121  SOSXLTC12AF72A7F54     23   5.0


Here is the recommendation for the userId: 53

      userID             song_id  score  Rank
378       53  SOBONKR12A58A7A7E0     34   1.0
209       53  SOAUWYT12A81C206F1     31   2.0
1399      53  SOFRQTD12A81C233C0     30   3.0
223       53  SOAXGDH12A8C13F8A1     24   4.0
4526      53  SOSXLTC12AF72A7F54     23   5.0
```

6.3. KNN based collaborative filtering

```
Out[33]: [Prediction(uid='A2V2URLB31HG59', iid='B00COEBCXY', r_ui=5.0, est=4.739130434782608, detail
         s={'actual_k': 1, 'was_impossible': False}),
          Prediction(uid='A2VODABWSVHV8E', iid='B0019UEY66', r_ui=5.0, est=4.265800582326734, detail
         s={'was_impossible': True, 'reason': 'User and/or item is unknown.'}),
          Prediction(uid='A376OJHLE6SU9Q', iid='B002N8A098', r_ui=5.0, est=4.391752577319588, detail
         s={'actual_k': 0, 'was_impossible': False}),
          Prediction(uid='A203OCQQ12MAVT', iid='B000MSZQ4K', r_ui=5.0, est=4.265800582326734, detail
         s={'was_impossible': True, 'reason': 'User and/or item is unknown.'}),
          Prediction(uid='A11ULAZ4B5ZXC8', iid='B00005A1K1', r_ui=5.0, est=4.909090909090909, detail
         s={'actual_k': 0, 'was_impossible': False}),
          Prediction(uid='A1F1A0QQP2XVH5', iid='B0079TQFQQ', r_ui=5.0, est=3.9597563526555715, detai
```

```
: # get RMSE
  print("User-based Model : Test Set")
  accuracy.rmse(test_pred, verbose=True)

  User-based Model : Test Set
  RMSE: 0.9401

: 0.9401136453139496
```

6.4. Collaborative filtering using Autoencoders

```
Namespace(path_to_predictions='preds.txt', round=False)
####################
RMSE: 0.9746437597050387
####################
```

7. Conclusion

In this article I have explored various traditional approaches of machine learning recommendation systems models as well started to explore how reinforcement learning can be used to build collaborative filtering-based recommendation systems. Further I would like to explore different dataset with audio and image data to build recommendation using deep learning and Reinforcement learning.